5J-03

マルチGPU上での深層畳み込み敵対的生成ネットワークの階層的並列処理

Hierarchical Parallelization of Deep Convolutional GAN on Multi-GPU

根本　祐輔†　　吉田 明正†
Yusuke Nemoto　　Akimasa Yoshida

1　はじめに

深層畳み込み敵対的生成ネットワーク（DCGAN）は，敵対的生成ネットワーク（GAN）の一種であり，データの特徴を学習することで実在しないデータを生成することができる．DCGANの生成ネットワークと識別ネットワークには畳み込みが用いられており，より精度の高い画像を生成できるが，実行時間が長くなるという問題がある．本稿では，マルチGPU上でタスク並列とデータ並列を使用し，DCGANの階層的並列処理を実現する手法を提案する．本手法はOpenMPとCUDAを用いて実装されており，NVIDIA RTX A5500を4台搭載したXeonサーバ上での性能評価の結果，ネコのカラー画像生成の実行時間は大幅に短縮されており，提案手法の有効性が確認された．

2　敵対的生成ネットワーク

本章では，敵対的生成ネットワーク（GAN）並びに深層畳み込み敵対的生成ネットワーク（DCGAN）について述べる．

2.1　GAN

GAN（Generative adversarial networks）とは，生成器と識別器ネットワークモデルが競い合うように学習する生成AIの一種である．生成器は，標準正規分布で定義されたランダムなノイズを入力として偽物のデータを生成する．識別器は，生成器で生成した偽物のデータと本物のデータを入力とし，本物か偽物かを数値でそれぞれ出力する．生成器と識別器のそれぞれで逆伝播で勾配を求める．パラメータを更新していき，より誤差のないデータを生成するモデルである．

2.2　DCGAN

DCGAN（Deep Convolutional GAN）とは，GANの生成器と識別器の2つのネットワークに畳み込みを組み込んだものである．通常の全結合のGANでは，各ニューロンが独立して，隣り合うピクセル間の関係性を埋め込むことができないことが欠点である．そこで，畳み込みニューラルネットワークを使用することで学習の精度を向上させる．しかし，層が多層化したことからGANよりも学習の時間が長くなるという問題が新たに発生する．本稿では，マルチGPUを用いて2つのネットワークを並列に実行し，さらにデータ並列を適用し，高速化を実現する．先行研究として，畳み込みニューラルネットワークモデルの高速化[1]やDCGANのモデル軽量化などが提案されている[2][3]．

†明治大学大学院先端数理科学研究科ネットワークデザイン専攻
Network Design Program，Graduate School of Advanced Mathematical Sciences，Meiji University

```
#pragma omp parallel
{
   #pragma omp private(p)
   for (int ep =0; ep < 300; ep++){
      for (int id =0; id < im/b; id++){
         p=omp_get_thread_num();
         if(p==0){
            バッチサイズの前半実行(生成器);
            生成器の勾配をホストに転送;
            pthread_cond_broadcast(a);
         }else if (p==1){
            バッチサイズの前半実行(識別器);
            識別器の勾配をホストに転送;
            pthread_cond_broadcast(b);
         }else if (p==2){
            バッチサイズの後半実行(生成器);
            pthread_cond_wait(a);
            ホストから勾配を受け取る;
            勾配を平均化;
            生成器のパラメータを更新;
            パラメータを各GPUに転送;
         }else if (p==3){
            バッチサイズの後半実行(識別器);
            pthread_cond_wait(b);
            ホストから勾配を受け取る;
            勾配を平均化;
            識別器のパラメータを更新;
            パラメータを各GPUに転送;
         }
         #prgma omp barrier
      }
   }
}
```

図1　階層的並列処理の部分コード．

3　DCGANのタスク並列とデータ並列

本章では，DCGANをデータ並列とタスク並列を組み合わせた階層的並列処理で実行する方法について述べる．

3.1　DCGANのタスク並列

GPU上でプログラムを実行するためにCUDAで実装を行う．CUDAコードでは，カーネル関数ごとに最適なブロック数とスレッド数を決めることで高速化を図る．畳み込み部分は計算時間の大半を占めるため，処理を複数に分けて実行し速度向上を実現する．また，CUDAカーネルで書かれたコードは，OpenMPを使って2スレッド生成され，2GPUで実行される．

3.2　DCGANのデータ並列

通常のバッチサイズを2分割し，2つのGPUでそれぞれ実行する．実行するバッチサイズが半分になることで，計算量が減る．パラメータを更新するときは，1つのGPUに勾配を集め，平均化し，更新する．

表 1　性能評価に用いるマシン.

CPU	Intel Xeon Gold 6326 (2.90GHz 16 Core)
GPU	NVIDIA A5500 24GB
メモリ	DDR4-3200 REG ECC 32GB(256GB)
OS	Ubuntu 20.04LTS
CUDA	10.2.89

	出力	パラメータ
入力ノイズ	100*1*1	
転置畳み込み	256*4*4	100*256*4*4
バッチ正規化	256*4*4	256, 256
ReLU関数	256*4*4	
転置畳み込み	128*8*8	256*128*4*4
バッチ正規化	128*8*8	128, 128
ReLU関数	128*8*8	
転置畳み込み	64*16*16	128*64*4*4
バッチ正規化	64*16*16	64, 64
ReLU関数	64*16*16	
転置畳み込み	32*32*32	64*32*4*4
バッチ正規化	32*32*32	32, 32
ReLU関数	32*32*32	
転置畳み込み	3*64*64	32*3*4*4
Tanh関数	3*64*64	

(a) 生成器ネットワーク

	出力	パラメータ
入力	3*64*64	
畳み込み	32*32*32	32*3*4*4
LReLU関数	32*32*32	
畳み込み	64*16*16	64*32*4*4
バッチ正規化	64*16*16	64, 64
LReLU関数	64*16*16	
畳み込み	128*8*8	128*64*4*4
バッチ正規化	128*8*8	128, 128
LReLU関数	128*8*8	
畳み込み	256*4*4	256*128*4*4
バッチ正規化	256*4*4	256, 256
LReLU関数	256*4*4	
畳み込み	1*1*1	1*256*4*4
Sigmoid関数	1*1*1	

(b) 識別器ネットワーク

図 2　DCGAN のモデル.

3.3　DCGAN の階層的並列処理

マルチ GPU 上でタスク並列とデータ並列を使用し，DCGAN の階層的並列処理を実現する手法を提案する．DCGAN で階層的並列処理を行うコードを図 1 に示す．im は学習画像の合計，bat はバッチサイズ，a と b は同期変数とする．GPU0 と GPU1 がバッチサイズ前半の実行，GPU2 と GPU3 が後半の実行を行うようにデータ並列を適用している．そして，GPU0 と 1，GPU2 と GPU3 でそれぞれタスク並列を適用している．パラメータの更新は，生成器は GPU2，識別器は GPU3 で行う．同期は pthread_cond_wait() を使用し，バッチ前半の勾配を受け取った後にパラメータ更新を行う．

4　マルチ GPU 上での DCGAN の性能評価

本稿では，300 エポックで実行を行い，トレーニング画像としてネコのカラー画像 2944 枚を使用する．バッチサイズは通常を 64 とし，データ並列を用いる場合には，1 モデルを 32 バッチサイズで実行を行う．CuPy[4]，1GPU の CUDA，2GPU でタスク並列 [5] とデータ並列，4GPU では階層的並列処理により評価を行う．1 エポックの処理時間，バッチ単位での処理時間，速度向上比を確認する．オリジナルモデルの学習の精度を見るために元のモデルと比較した損失の推移の評価も行う．入力は，100 要素の標準正規分布をノイズとし毎回バッチランダムに生成している．損失誤差には，バイナリークロスエントロピーを使用し，最適化手法には Adam を使用している．

なお，本性能評価に用いるマシンの環境を表 1 に示す．DCGAN のモデルは図 2 に示す．

4.1　DCGAN の階層的並列処理の性能評価

提案する CUDA 実装による階層的並列処理の実行時間は，表 2 に示す通りである．デバイスで必要なデータは，事前にホストから転送しておく．訓練画像ファイル読み込みとホストからデバイスへのデータ転送に要する時間は，1.012[s] であり，300 エポックの実行時間に比べると小さい．

1 エポックによる実行時間をみると，4GPU の実行では 2.184[s] であった．GPU 上で実行するためのライブラリである CuPy と比べて 6.558[倍] の速度向上を得ることができた．先行研究である CUDA のタスク並列の実行時間が 3.197[s] であり，それに比べて 1.463[倍] の速度向上を得ることができた．通常の 1GPU の実行と比較すると，2.811[倍] の速度向上を得ることができた．データ並列 2GPU の実行と比べて 4GPU での実行では，1.798[倍] の速度向上を得ることができた．

表 2　RTX 5500 上での DCGAN の実行時間.

実行形式	CuPy	CUDA	CUDA	CUDA	CUDA
GPU 数	1	1	2	2	4
GPU 間並列処理	なし	なし	タスク	データ	階層的
1 エポックの処理時間 [s]	14.324	6.140	3.197	3.927	2.184
速度向上比 [倍]	1.000	2.333	4.480	3.649	6.558
バッチ単位での処理時間 [s]	0.311	0.133	0.070	0.085	0.047

4.2　DCGAN の損失の評価

損失の推移を見ることで生成器と識別器の学習の度合いを確認することができる．生成器の損失は，生成された画像が本物に近いと判定すれば損失は小さくなる．識別器の損失は，偽物の画像を入力として判定した損失と本物の画像を入力として判定した損失の合算である．損失が小さいほど偽物と本物の判別ができている．CuPy，階層的並列処理を用いた CUDA 実装における生成器と識別器の損失推移は同等であった．これらの結果から，CuPy 実行と階層的並列処理実装のモデルに性能の差はなく，提案手法の有効性が確認できる．

5　おわりに

本稿では，マルチ GPU 上での深層畳み込み敵対的生成ネットワークの階層的並列処理を提案した．DCGAN を OpenMP/CUDA により実装し，タスク並列とデータ並列を合わせた並列性を実現した．RTX5500 搭載 Xeon サーバで性能評価を行ったところ，4GPU での階層的並列処理は，CuPy 実行と比べて 6.558[倍]，1GPU の CUDA 実行と比べて 2.811[倍] の速度向上を得ることができた．以上の結果から，提案手法の有効性が確認された．

参考文献

[1] Muhammadjon Musaev, Mekhriddin Rakhimov. Accelerated Training for Convolutional Neural Networks, 2020 International Conference on Information Science and Communications Technologies (ICISCT), 2020.

[2] Haoyuan Chi, Zebin Zhang, Qiqi Ge, Wenhuan Zhu. Infrared Image Colorization Algorithm Based on DCGAN and Its Edge Device Acceleration, 2022 IEEE International Conference on Civil Aviation Safety and Information Technology (ICCASIT), 2022.

[3] Aswathy Ravikumar, Harini Sriraman. Single Node Acceleration of Generative Adversarial Networks using HPC for Image Analytics, CIIS '22: Proceedings of the 2022 5th International Conference on Computational Intelligence and Intelligent Systems, Pages 54-59, 2022.

[4] pometa0507, DCGAN-Numpy, https://github.com/pometa0507/DCGAN-Numpy, 2020.

[5] 根本祐輔，吉田明正．マルチ GPU 上での CUDA 実装による深層畳み込み敵対的生成ネットワークの並列処理，第 22 回情報科学技術フォーラム，2023.